Vol. 37 No. 1 情報処理学会論文誌 Jan. 199

空間分割モデルを用いた形状モデラ

米川和利† 小堀研一† 久津輪敏郎†

本論文では，工業デザイン分野における製品形状をモデリングする方法として，従来から行われていた発泡スチロールや粘土をへら等で盛る，削るといった作業を計算機上で実現できる形状モデラについて報告する．また，このモデラを実現するアプローチとして，形状モデルのための新しいデータ構造を提案する．現在CADで主流のデータ構造であるB-Repsは幾何要素をデータ構造に持っているため表示が速く，局所的な変形操作も可能であるため多くの3次元ソリッドモデラに採用されている．しかし，B-Repsは構成する要素が面，稜線，頂点であるために，定義された形状モデルに対する再加工を行う操作には，設計者はそれらの要素を意識して操作を行わなければならず，またB-Repsでの立体間の集合演算は計算機処理負荷が大きくなるという問題点もあった．そこで形状モデルとしてデータ構造の簡単な空間分割モデルを用いることが考えられるが，このモデルでは形状表現精度を上げると構成するキューブ数が膨大になるという問題点があった．そこで本論文では空間を階層的に管理するOctreeデータ構造に表現精度を上げる拡張を加えることにより，キューブ数の増加を抑えて，自然な操作で形状の加工が行える工業デザインのためのモデラを実現した．

A Geometric Modeler by Using Spatial-Partitioning Representations

KAZUTOSHI YONEKAWA,† KEN-ICHI KOBORI† and TOSHIRO KUTSUWA†

This paper describes a new modeler that can handle a computer model like a clay modeling and proposes new data structure in order to realize the modeler. In the field of CAD, B-Reps is major data structure because local operation is easy and the rendering speed is fast relatively. However, it is necessary for a designer to be conscious of faces, edges and vertices of a shape explicitly in creating a B-Reps-model and its Boolean set operation requires much computational time. On the other hand, the designer can have natural operation of a geometry shape by spatial partitioning representations because of its simple data structure. This data structure has a problem that numbers of cube are necessary in high resolution representation. In order to solve this problem, we propose a data structure using extended octree and realize a new modeler that can operate the shape in natural and real-time.

1. まえがき

現在，製品形状のデザインでは，設計者が発泡スチロールや粘土をカッター，へら等で盛る，削るといった作業が行われているが，この作業を計算機上で実現することは，製品データのコンピュータ化，シミュレーション，CAMへの利用等，製品設計の向上につながり，設計の効率化につながる．

この作業を計算機上で実現するため，形状モデリングのインタフェースを現実の操作に近づける試みがなされており，様々なモデラが提案されているが[1),2)]，一般に特殊なハードウェアが必要となり利用は容易ではない．そこで我々はCADにおけるインタフェ
はモデルのデータ構造と密接な関係があると考え
ンタフェースを実際の操作に近づけるのではなく
実に近いモデルを計算機内の形状モデルのデータ
に導入することによって，実際の操作に近似した
ローチを試みた．

ところで3次元CAD分野において，ソリッド
ルは立体を完全に表現することができ，集合演算
能であるため，有効なデータ構造であることが知
ている[3)]．特にB-Repsは実際の幾何要素をデー
造に持っているため表示が速く，局所的な変形操作
可能であるので，多くの3次元モデラに採用され
る．しかし，B-Repsを用いたモデラではデザイ
形状の変形操作を行う際に，モデラのデータ構造
なわちB-Repsの構成要素である面・稜線・頂点

† 大阪工業大学
Osaka Institute of Technology

識しなければならず，モデルのデータ構造に対する理解と慣れが必要である．また，形状変形で頻繁に使われる立体間の集合演算も計算機への負荷が大きいため対話的な操作には向いていない．

そこで我々は，現実の物体形状が無数の粒子の集合体で構成されているのと同様に，計算機上のモデル形状も微細な立方体（Voxel）の集合体で構成することができれば，モデリング操作を現実の操作に近づけることが可能であると考えた．近年，このようなボリュームグラフィクスがモデリングの手法として注目されている[4),5)]．Voxelは形状の面，稜線，頂点を意識する必要がないので形状を大局的にイメージするデザイナに適しては初期の外観形状を決定するための最適な形状モデルであると考えられる．

しかし，Voxelをデータ構造として持つためには膨大なメモリ空間と処理速度が必要となり，現状のワークステーションではハードウェアの処理能力に限界がある．そこで，モデラのデータ構造にVoxelよりも精度よく形状を表現することができるデータ構造を導入することによって，ソフトウェア的な解決を試みた．

本論文では，データ構造に拡張を加えたOctreeデータ構造（以後，Octree）を用いたモデラ（以後，Octant-Modeler）を提案する．デザイナはこのモデラを使用することによりデータ構造を意識することなく自然な操作で形状を操作することが可能となる．

本論文では，まず，従来のOctree[6),7)]の問題点を指摘する．次にこの問題点を解決するための拡張を加えたOctreeを提案し，このデータ構造による集合演算のアルゴリズムを示し，実際にモデル形状を作成してその効果を検証する．

2. データ構造

これまでOctreeをデータ構造に用いた形状モデラの研究は報告されているが[8)]，後述する様々な問題が生じるため，本論文で提案するOctant-Modelerのデータ構造には，従来のOctree（以後，Std-Octree）に拡張を加えたデータ構造（以後，Ex-Octree）を用いた．この章では，まず最初にStd-Octreeについて簡単に述べ，このデータ構造をモデラに用いることの長所と短所を述べる．次にStd-Octreeの問題点を解決したEx-Octreeについて詳細を述べる．

2.1 Std-Octree データ構造

Std-Octreeとは，物体形状に対して完全に内部であるBLACK，完全に外部であるWHITE，さらに分割されているMIXの3状態（以後Octant状態）に空間を再帰的に分割することで形状を表現するデータ構造である．Std-Octreeで構成された形状の例を図1 (a)に示し，ツリーの状態を同図(b)に示す．同図は2次元のQuadtreeで示しているが，実際の処理は3次元であるからOctreeとなる．以後の図は簡略化のため基本的には2次元で表現することとする．

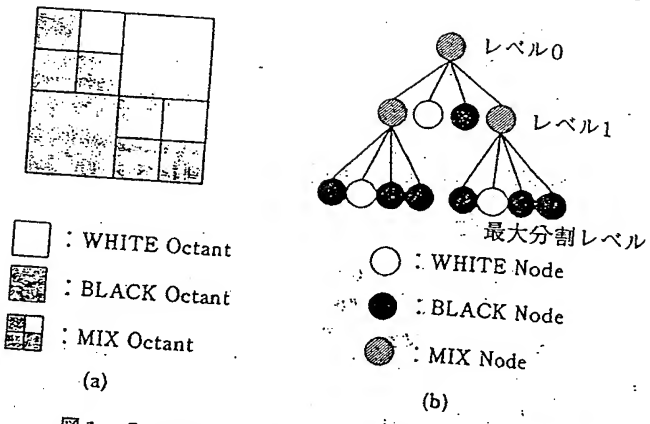


図1 Std-Octree表現．(a) 形状表現例，(b) 木構造．
Fig. 1 Std-Octree representation. (a) An example of shape representation, (b) Tree structure.

本論文では，ツリーの深さをレベルと呼び，ツリーのノードに対応する立方体をOctantと呼ぶ．したがって，ツリーのルートノードをレベル0のOctant，ルートノードからn回分割が行われたノードをレベルnのOctantと呼び，ツリーの親ノード・子ノードの関係と同様に，Octantについても，親Octant・子Octantと呼ぶことにする．また，Octantの分割を無限に繰り返すことが可能であれば空間分割モデル固有の問題であるエイリアシングは発生しなくなるが，計算機のメモリには制限があるため，さらに分割を繰り返さないレベルを設定しておき，これを最大分割レベルと呼ぶ．

2.2 Std-Octreeの長所と短所

Std-Octreeをモデラに用いることによる長所は以下の点である．

(1) 形状の構成要素はVoxelと同様に立方体であるであるため処理が単純になる．

(2) B-RepsやCSGは形状処理時間および処理アルゴリズムが形状の複雑さに依存するのに対し，空間分割モデルであるStd-Octreeは，形状処理時間は表現精度に依存し，処理アルゴリズムは形状の複雑さに依存しない．

(3) 工業製品のような形状であれば，同じ空間分割モデルであるVoxelよりも少ないOctant数で形状を表現することができる．しかし，Std-Octreeをモデラに用いると以下のような問題点が生じる．

(1) Std-OctreeはVoxelよりも構成Octant数が少

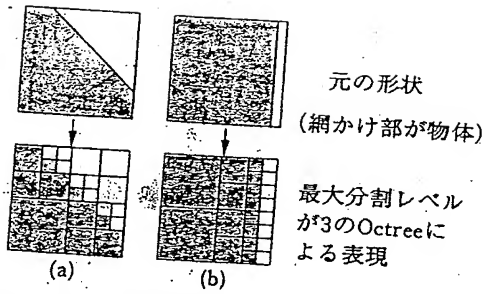


図2 Std-Octree の精度の問題. (a) 斜面の形状の場合, (b) 形状が Octant の境界に位置しない場合
Fig. 2 Accuracy problem of Std-Octree. (a) In case of representing a slope, (b) In case of passing through octants in vertical face.

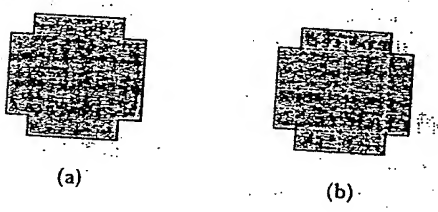


図3 Std-Octree でレンダリングするポリゴン. (a) 表面形状のポリゴン, (b) 形状内部のポリゴン (実線はレンダリングが必要なポリゴン, 点線はレンダリングが不必要なポリゴン)
Fig. 3 Polygons to be rendering. (a) Polygons on the surface, (b) Polygons inside of a model.

ないという長所を持つが, 図2(a)のように, 構成しようとする物体形状の表面が, X, Y, Z 各軸のいずれにも垂直でない斜面を含む場合や, 同図(b)のように垂直であっても物体形状の表面の座標が分割される Octant の境界に位置しない場合は, Octant の分割を繰り返さなければならず, 製品形状を表現する場合, 多くの形状の表面は最大分割レベルにまで分割されてしまう. したがって, 精度を上げて形状を表現しようとすると, 形状を構成する Octant 数が急増してしまう.

(2) 対話的なインタフェースを実現するにはモデルのレンダリングを実時間で処理することが重要である. これをグラフィックスワークステーション上で実現するためには BLACK-Octant を 6 枚の正方形ポリゴンで構成されている立方体として, ジオメトリックエンジンを用いてレンダリングする必要がある. しかし, 図3(a)の実線で示すポリゴンのように実際はモデル形状の表面を構成するポリゴンのみをレンダリングするだけで十分であるのに対し, BLACK-Octant を構成する正方形ポリゴンが形状の表面であるか内部であるかの情報を持たないために, 図3(b)の点線で示す形状内部のポリゴンについてもレンダリングする必要があり, 処理負荷が大きくなる.

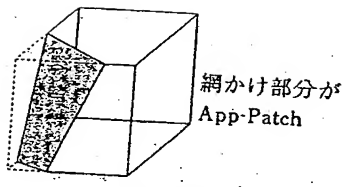


図4 PATCH-Octant
Fig. 4 PATCH-Octant.

2.3 Ex-Octree 構造

上記の問題を解決するために Octree に B-Reps データを保持するハイブリッド型のモデルが提案されている[9]が, Octant に正確な B-Reps の頂点, 稜線情報を保持するため, 空間分割モデルの単純性が失われ高速な集合演算処理が期待できない. そこで本論文で提案する Ex-Octree では, Std-Octree の BLACK, WHITE, MIX の Octant 状態に Marching-Cubes 法で提案されているパッチ[10]を保持できる図4のような構造を持った Octant を導入した. 以後, この Octant を PATCH-Octant, 張られたパッチを App(Approximate)-Patch と呼ぶことにする. この PATCH-Octant を追加することにより, 従来, 図3のような Octant の分割を繰り返すことにより表現していた形状を 1 つの PATCH-Octant で表現することが可能となる. また, App-Patch はモデル形状の表面にのみ張られるため, 図3(b)のようにモデル形状の内部にあるポリゴンをレンダリングすることはない. Marching-Cubes 法は表示のための手法であり, 与えられた離散的な濃度データから表示用のパッチを求めるだけであるので Voxel の 8 頂点にデータを持つだけであるが, 形状モデルとして用いるためには幾何情報を保持する必要がある. したがって, PATCH-Octant には Octant を構成する稜線 12 本それぞれに −1.0 〜+1.0 までの符号付きの実数値を情報として持つ App-Patch を表現する. 以後, それぞれの稜線の数値を ED データ (Edge Distance Data), PATCH-Octant が保持する 12 個の ED データを App-Patch データと呼ぶことにする.

任意の App-Patch に対する App-Patch データを求める方法を以下に述べる.

(1) Octant の各稜線についてその端点の各 X, Y, Z の値の小さい方を始点, 大きい方を終点とした方向を持たせる. また, 各稜線に図5に示すように番号を割り当てる.

(2) Octant を構成する頂点がモデル形状の内部であるか外部であるか, および, 始点から稜線上の App-Patch を構成する頂点までの距離 (L) によって PATCH-Octant の ED データを表1により決定する

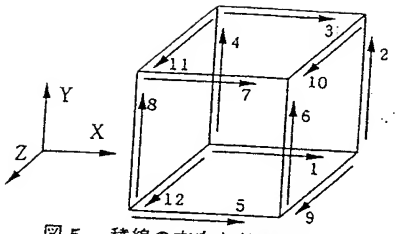


図5 稜線の方向と稜線番号
Fig. 5 Edge direction and its number.

表1 頂点の状態による ED データの決定
Table 1 Deciding ED-data based on vertex status.

始点の状態	終点の状態	ED データ
内側	外側	L
外側	内側	−L
内側	内側	1.0
外側	外側	−1.0

表2 図4の状態での ED データ
Table 2 ED-data in Fig.4.

稜線番号	始点	終点	L	ED データ
1	内部	内部	−	1.0
2	内部	内部	−	1.0
3	内部	内部	−	1.0
4	内部	内部	−	1.0
5	外部	内部	0.2	− 0.2
6	内部	内部	−	1.0
7	外部	内部	0.75	− 0.75
8	外部	外部	−	− 1.0
9	内部	内部	−	1.0
10	内部	内部	−	1.0
11	内部	外部	0.5	0.5
12	内部	外部	0.9	0.9

たとえば図4の場合は表2のような ED データが得られる．

3. 形状加工処理

3.1 Octant-Modeler のモデリング概要

Octant-Modeler では被加工形状に Ex-Octree，加工形状に凸型 B-Reps を用い，Ex-Octree との集合演算を繰り返すことによって，最終的に目的とする形状を得る．集合演算には，和・差の演算（以後，演算モード）が可能である．加工道具として任意の凸形状を定義することができる．すなわち，デザイナが用いる発泡スチロールが Ex-Octree に，ヘラやヒートカッタ等の道具が B-Reps 等で表現される．本論文では道具として凸形状の B-Reps を扱ったが，凹形状を凸形状に分割することによって凹形状での集合演算も可能である．

3.2 最大分割レベルにおける形状処理

図6 (a) に示す PATCH-Octant に同図 (b) のような

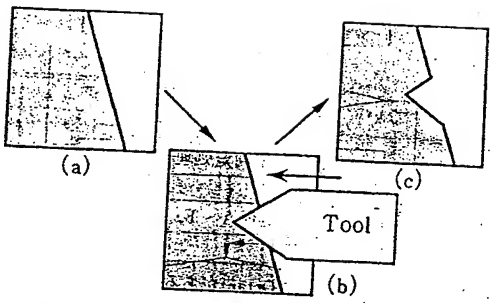


図6 PATCH-Octant で表現できない形状
Fig. 6 An example of a shape that can't be represented by PATCH-Octant.

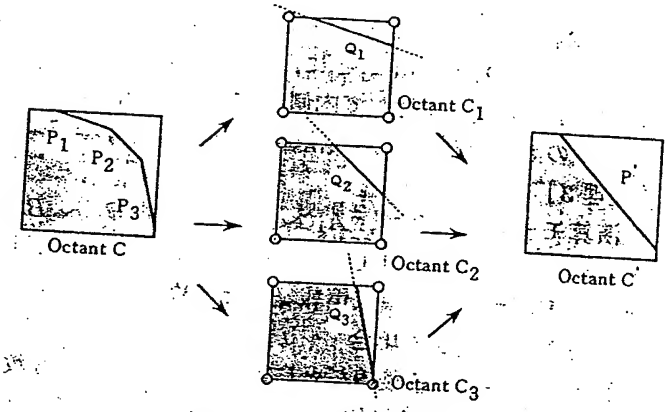


図7 App-Patch 変換処理
Fig. 7 Transformation of App-Patch.

道具形状による集合演算が行われると，同図 (c) の形状となるが，同図 (c) の形状は2章で述べた PATCH-Octant では表現することはできない．したがって，3.3節で詳述する処理によって，この PATCH-Octant を子 Octant に分割し，それぞれの子 Octant が分散して形状を保持することにより全体の形状を表現する．ここでは，分割を繰り返すことによって，Octant の分割レベルが最大分割レベルに達しても，Octant 内に存在する形状を App-Patch で表現できない場合に，App-Patch に変換する方法について説明する．App-Patch 変換処理は変換する前の元の形状から，差によって App-Patch を求めるか，和によって求めるかによって若干手順が異なる．以下に処理の手順を示す．

[処理1]

図7に示すように PATCH-Octant C 内に存在する形状を構成する n 枚のパッチをそれぞれ $P_1 \sim P_n$ とすると，パッチ P_i を含む無限平面 Q_i それぞれ1枚ずつを App-Patch とする PATCH-Octant $C_1 \sim C_n$ を考え，その App-Patch データを $A_1 \sim A_n$ とする．図7は $n=3$ の場合である．なお，以下の E_{ij} とは A_i の稜線番号 j $(j=1,\cdots,12)$ の ED データを示す．

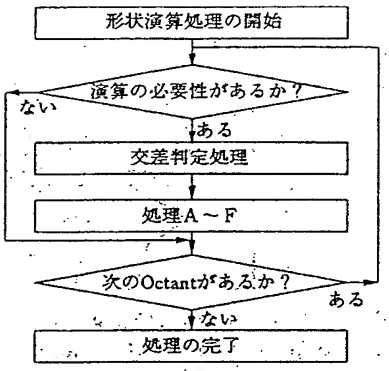


図8 形状演算処理の流れ
Fig. 8 A general flow of boolean set operation.

[処理2]

$C_1 \sim C_n$ の Octant ごとに，C_i を構成するそれぞれの8頂点が Q_i に対して内側'1'であるか外側'0'であるかの8ビット長の頂点情報 B_i を求める．

[処理3]

演算モードに応じて，$A_1 \sim A_n$ および $B_1 \sim B_n$ から以下の処理により Octant C' の App-Patch に対応する ED データ，頂点情報 B' を求める．

・演算モードが差の場合

(1) $A_1 \sim A_n$ の各稜線ごとに数値を比較して最も数値の小さい値を，最終的に求める App-Patch データである A' の各稜線に代入する．すなわち，

$$E'_j = \mathrm{Min}(E_{ij}) \qquad (i = 1, \cdots, n,\ j = 1, \cdots, 12)$$

(2) $B_1 \sim B_n$ の AND ビット演算の結果として B' を求める．

$$B' = \mathrm{AND}(B_1, \cdots, B_n)$$

(3) それぞれの稜線ごとに，稜線の始点と終点の頂点情報を調べ，どちらも0の場合は E'_j の値を -1.0 とする．

・演算モードが和の場合

(1) 差の場合と同様に，$A_1 \sim A_n$ の各稜線ごとに数値を比較し，最も数値の大きい値を A' の各稜線に代入する．すなわち，

$$E'_j = \mathrm{Max}(E_{ij}) \qquad (i = 1, \cdots, n,\ j = 1, \cdots, 12)$$

(2) $B_1 \sim Bn$ の OR ビット演算の結果として B' を求める．

$$B' = \mathrm{OR}(B_1, \cdots, B_n)$$

(3) 稜線の始点と終点の頂点情報を調べ，どちらも1の場合は E'_j の値を $+1.0$ とする．

3.3 Octree と B-Reps との集合演算処理

Octree と B-Reps との立体集合演算は以下の (1)～(3) の順に処理する．また，この処理の概略を図8に示す．

(1) Octant 状態が BLACK で演算モードが和の場合，あるいは Octant 状態が WHITE で演算モードが差の場合は集合演算を行う必要がないために，以下の (2) 以降の処理は行わず次の Octant に処理を移し，(1) の再帰処理を行う．

(2) Octant と B-Reps のパッチ群との交差判定を行う．交差判定方法は Octant の8頂点とパッチを含む無限平面との符号付きの距離を求めて1点でも異符号となる距離があれば，交差の可能性があるとして対象 Octant とパッチを X-Y，Y-Z および Z-X 平面に投影した2次元での正確な交差判定を行う[11]．判定結果の Octant とパッチ群との交差状態は図9に示す外側（Outside），内側（Inside），パッチが1枚交差（Single），パッチが複数枚交差（Multiple）の4状態である．

(3) Octant 状態とパッチ群との交差状態により表3に示す A～F の6状態に分類し，それぞれの状態に応じて以下の処理を行う．

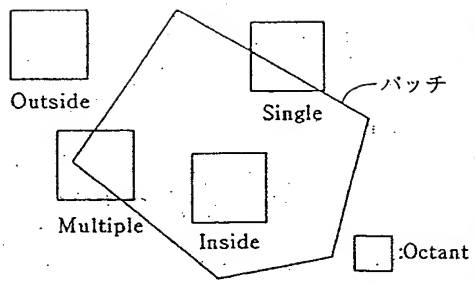


図9 Octant とパッチの交差状態
Fig. 9 Interference between octant and patch.

[処理A] Octant がパッチ群に対して完全に内側に存在する状態

(1) Octant が MIX-Octant であれば，その子 Octant をツリーから削除する．

(2) 演算モードが和であれば Octant を BLACK Octant に，差であれば WHITE-Octant にする．

(3) 次の Octant に処理を移す．

[処理B] Octant がパッチ群に対して完全に外側に存在する状態

集合演算の対象とはならない領域なので Octant はそのままであり，次の Octant に処理を移す．ただし Octant が MIX-Octant であれば，その子 Octant はすべてパッチ群に対して完全に外側に存在する状態となるので，子 Octant は処理する必要がない．したがって，同じ親を持つ Octant へと処理を移す．

[処理C] BLACK/WHITE-Octant がパッチ群の中の1枚だけに交差している状態

(1) 交差パッチの App-Patch データを求める．

表3 形状演算処理の分類
Table 3 Classification of process in boolean set operation.

Octant 状態 \ 交差状態	Inside	Outside	Single	Multiple
BLACK/WHITE	A	B	C	F
MIX	A	B	F	F
PATCH	A	B	F	F
Max-BLACK/WHITE	A	B	C	D
Max-PATCH	A	B	E	E

Max は最大分割レベルを示す.

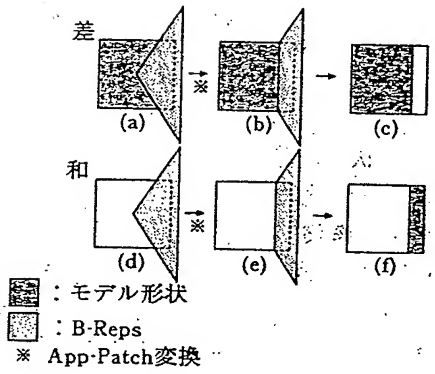


図10 処理Dの流れ
Fig. 10 Process D.

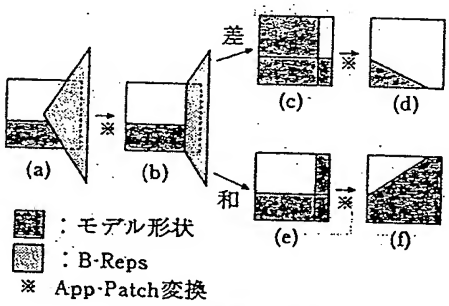


図11 処理Eの流れ
Fig. 11 Process E.

(2) 演算モードが差の場合，App-Patch の ED データの値の符号をすべて反転する.

(3) Octant を交差パッチが App-Patch となる PATCH-Octant にする.

(4) 次の Octant に処理を移す.

[処理 D] 最大分割レベルの BLACK/WHITE-Octant が，複数枚のパッチと交差している状態

(1) 図 10 (a), (d) に示すように，それぞれの交差パッチを $P_1 \sim P_n$ とし（同図は $n=2$），演算モードを差として前述の App-Patch 変換処理により，同図 (b), (e) に示すように複数枚のパッチから 1 つの App-Patch を求め，これを交差パッチとする.

(2) 演算モードが差の場合，図 10 (c) に示すように App-Patch の ED データの値の符号をすべて反転する.

(3) 図 10 (c), (f) に示すように Octant を求めた App-Patch を保持する PATCH-Octant とする.

(4) 次の Octant に処理を移す.

[処理 E] 最大分割レベルの PATCH-Octant がパッチと交差している状態

(1) 図 11 (b) に示すように [処理 D] と同様に，複数枚の交差パッチに対する App-Patch を求め，これを App-Patch データ A_1 とする.

(2) 演算モードが差の場合，図 11 (c) に示すように A_1 の ED データの値の符号をすべて反転したものを A_1 とする.

(3) PATCH-Octant が持つ App-Patch データを A_2 とする.

(4) A_1 と A_2 から 1 つの App-Patch を求めるために，$n=2$ として前述の App-Patch 変換処理により図 11 (d), (f) に示すように最終的に求める App-Patch を求める.

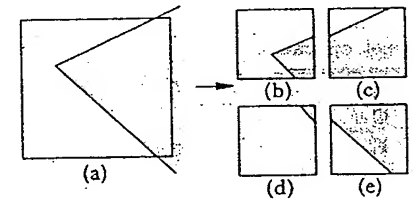
(5) Octant を求めた App-Patch を保持する PATCH-Octant とする.

(6) 次の Octant に処理を移す.

[処理 F] 処理 A～E 以外の状態

最大分割レベルではない Octant 内に複雑にパッチが交差しており，現在のレベルの Octant では精度が不十分なため形状を表現することができない．したがって，注目している Octant の処理を子 Octant に分割し，再帰的に形状演算処理を行う．すなわち，より高いレベルの Octant に処理を移すことにより，Octant の形状表現精度を上げ，個々の Octant の形状演算処理を簡単化する．この処理は形状加工を施される Octant 状態によって以下の 3 状態に分類して処理する.

- Octant が BLACK/WHITE-Octant の場合，すなわち図 12 に示すような WHITE-Octant（または BLACK-Octant）に B-Reps 形状が交差している場合は，この Octant を MIX-Octant とし，親 Octant と同じ状態を持つ子 Octant を現在の Octant のノードの下に作成し，8 個（図 12 では 2 次元のため 4 個）の子 Octant に処理を移す.
- Octant が MIX-Octant の場合は，Octant が持つ



(a)～(e) : BLACK/WHITE-Octant(図ではWHITE)

図 12 Octant 状態が BLACK/WHITE の場合
Fig. 12 In case of BLACK or WHITE-Octant.

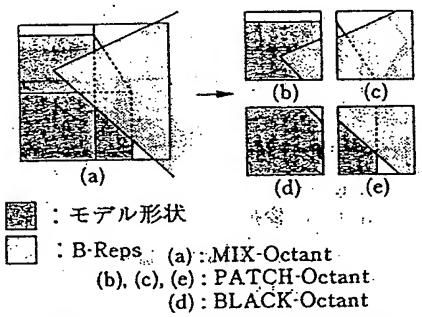


図 13 Octant 状態が MIX の場合
Fig. 13 In case of MIX-Octant.

子 Octant に処理を移す．図 13 では MIX-Octant が 2 枚のパッチと交差しているため，MIX-Octant の子 Octant に処理を移している例である．

• Octant が PATCH-Octant の場合は図 14 に示すように，この Octant を MIX-Octant とし，親 Octant の App-Patch を子 Octant に分割する．すなわち，親 Octant の App-Patch データより 8 個の子 Octant の App-Patch データを作成し，子 Octant に処理を移す．

4. 実験と考察

4.1 実験方法

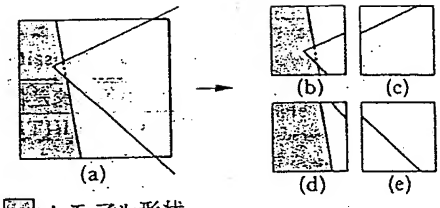
Ex-Octree と Std-Octree との集合演算速度および生成された Octant 数を比較するために次の実験 (1)～(3) を行った．実験では最大分割レベルを 5～8 に設定し，それぞれについて処理時間，Octant 数を測定した．ここでは本モデラが対象とする家電製品の意匠設計の初期段階での精度を考えて，分割レベルを 8 まで設定している．これは 25 cm^3 の形状に対して 1 mm の精度に相当する．なお，この実験に使用した計算機は SGI 社の Indy（R4400SC，150 MHz）である．

実験 (1) 形状の盛りつけ（4 角柱）

直方体の上面に B-Reps で構成された 4 角柱を加算する．

実験 (2) 形状の盛りつけ（12 角柱）

直方体の上面に B-Reps で構成された 12 角柱を加算する．



(a) : PATCH-Octant
(b), (d) : PATCH-Octant
(c), (e) : WHITE-Octant

図 14 Octant 状態が PATCH の場合
Fig. 14 In case of PATCH-Octant.

実験 (3) 形状の削除

直方体に球を減算する．

4.2 実験の考察

結果のグラフを図 15 に示し，生成されたモデル形状の写真を図 16～図 18 に示す．なお，実験 (1) では斜面の形状がないため，両者とも生成されたモデル形状はほぼ同一である．実験 (1)，(2) では，最大分割レベルが高くなるにつれて，処理時間，および Octant 数において Ex-Octree と Std-Octree との差が大きくなってきている．これは，Ex-Octree は最大分割レベルが高くなっても形状の角の部分のみが分割されるに対し，Std-Octree はモデル形状が存在するすべての境界部分において分割が行われるためである．

実験 (3) において Ex-Octree が Std-Octree に比べて最大分割レベルが 8 の場合，約 3.5 倍の処理時間を要している．これは球形状では集合演算が Ex-Octree も Std-Octree と同様にモデル形状のすべての境界部分で最大分割レベルまで行われるためであり，パッチを張るための App-Patch データを生成する処理の加があるためと考えられる．また，本論文で提案した Ex-Octree を形状モデルとし，3 章の形状加工処理による直方体，円柱，円錐，球等の集合演算を用いて実際の製品形状をモデリングした例を図 19 (a)，(b) に示す．

同図 (a) に対して Std-Octree では生成された Octant 数は 248889，Ex-Octree では 108089 であり，43%に減少している．同図 (b) では Std-Octree で 895649 に対して Ex-Octree では 163649 と 18%まで大幅に減少している．(a) の形状での Octant 数の減少が (b) の形状の減少率ほどではないのは，電話器の球面の一部を多用してしているため，本手法でも最大分割レベルまで分割が繰り返されるためである．実験 (1)～(3) および実際の製品形状のモデリングにより以下のことが明らかとなった．

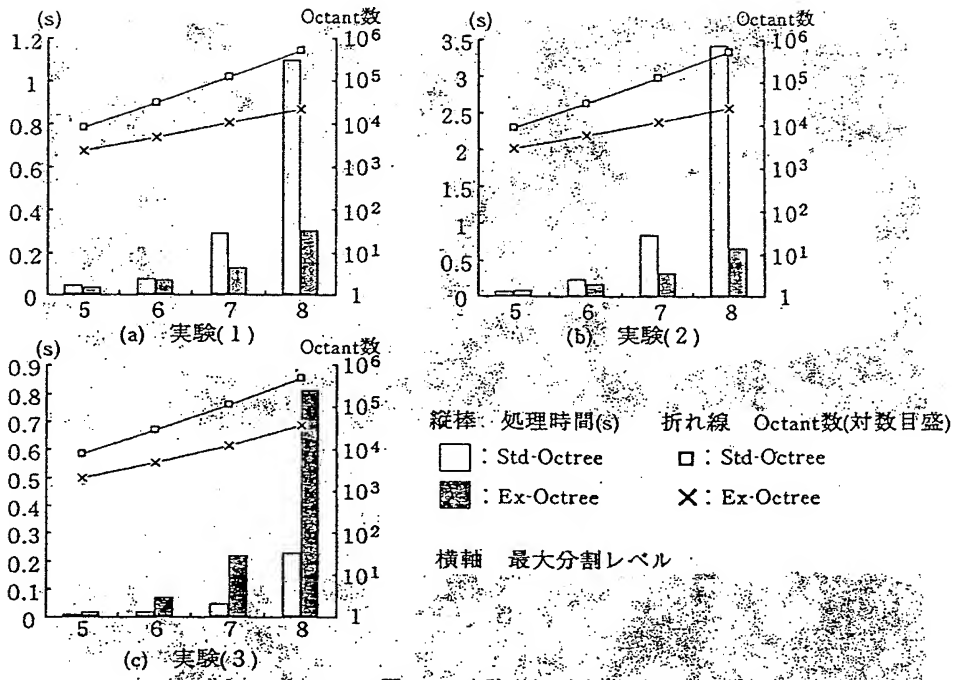


図15 実験(1)~(3)
Fig. 15 Experiment(1)~(3).

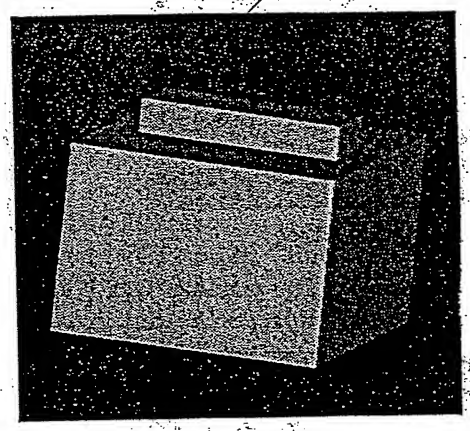


図16 実験(1)
Fig. 16 Experiment (1).

(1) 最大分割レベルをStd-OctreeとEx-Octreeを同レベルに設定した場合，球形状のようなすべての境界部分が最大分割レベルまで分割が繰り返される形状を除いては，Ex-Octreeの方が集合演算の処理時間が少なく，また，Octant数も少ない．さらにOctant数の減少によりレンダリング時間を短縮することが可能となった．

(2) 球形状のようにまったく平面を含まない表面で構成されている形状の場合には，Ex-Octreeは処理時間が長くなる傾向にある．

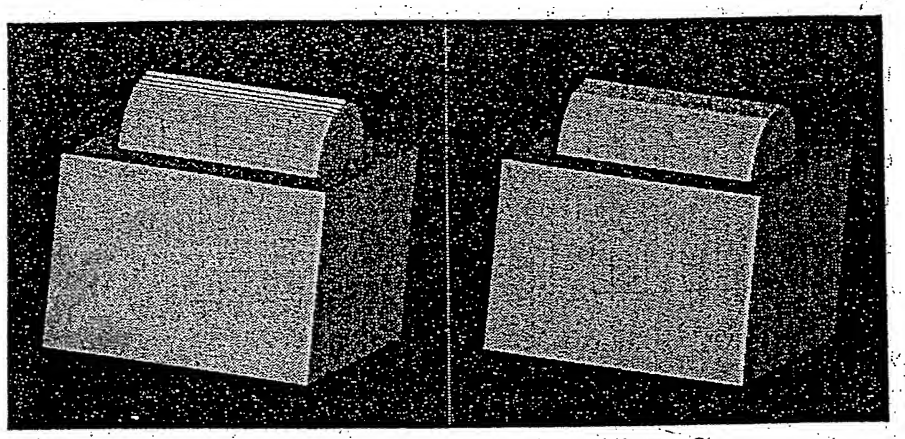
(3) 製品形状の図19(a)，(b)から明らかなようにEx-Octreeでは斜面の形状を滑らかに表現することが可能となった．

(4) 図19(a)，(b)の製品形状を作成するために行った個々の集合演算に要した時間はすべて1秒以内であり，対話的な加工を行うことが可能となった．

5. おわりに

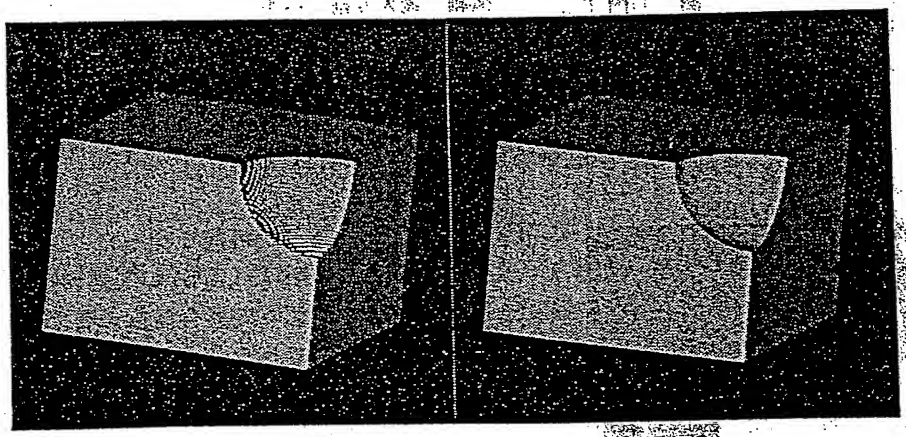
Ex-Octreeの導入により，従来のOctreeよりも表現能力が高まり，Octant数も激減した．これにより，デザイナにとって造形を行う形状モデルとして利用できる実用的なモデラを提案できた．しかし，実験結果でもわかるように，現在のところは最大分割レベル8までが一般的なワークステーションでの対話的な処理の限界である．今後，本モデラを適用できる家電製品形状の範囲を拡大するために，約50 cm^3の形状，すなわち最大分割レベルを9へあげることが課題となる．また，Ex-Octreeから従来よりCADに利用されているB-Repsへ変換する手法を開発して，既存アプリケーションの利用を図っていきたい．

謝辞 本研究の開発にあたってはシャープ(株)の工業デザイン部門の方々から貴重な御意見，御助言をいただきました．また，プログラム開発には研究室の亀井氏ほかの皆さんに協力をいただきました．ここに感謝いたします．



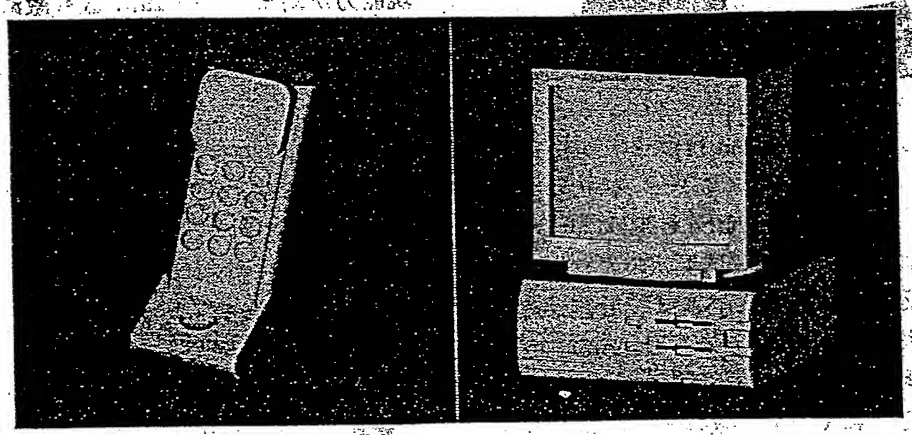
(a) Std-Octree　　(b) Ex-Octree

図17　実験(2)
Fig. 17　Experiment (2).



(a) Std-Octree　　(b) Ex-Octree

図18　実験(3)
Fig. 18　Experiment (3).



(a)　　(b)

図19　モデリング例
Fig. 19　Two examples of modeling.

参考文献

1) 高橋，金井，位守：クレイモデリング作業のための人工現実感システムの開発（第2報）仮想造形のデータ構造とアルゴリズム，精密工学会秋期大会学術講演会論文集，pp.15–16 (1993).
2) 平池，篠原：仮想作業スペースにおけるダイレクトモデリング手法，情報処理学会第43回全国大会講演論文集，4F-13 (1991).
3) 鳥谷，千代倉：3次元CADの基礎と応用，pp.11–18，共立出版 (1991).
4) 藤代，茅，國井：ボクセル指向3次元データ表現とその表示技術，情報処理学会誌，Vol.34, No.3, pp.285–298 (1993).
5) Galyean, T.A. and Hughes, J.F.: Sculpting: An Interactive Volumetric Modeling Technique, *Computer Graphics*, Vol.25, No.4, pp.267–274 (1991).
6) Foly J., van Dom A., Feiner S. and Hughes J.: *Computer Graphics Second Edition*, pp.533–562, Addison-Wesley (1990).
7) 登尾，福田，有本：オクトツリーを利用した3次元物体の最近点探索アルゴリズム，情報処理学会論文誌，Vol.30, No.3, pp.311–320 (1989).
8) 米川，小堀，久津輪：空間型インタフェースを用いたオクタントモデラ，情報処理学会第47回全国大会講演論文集，1V-10 (1993).
9) Brunet, P. and Navazo, I.: Solid Representation and Operation Using Extended Octrees, *ACM Transactions on Graphics*, Vol.9, No.2, pp.170–197 (1990).
10) Lorensen, W.E. and Cline, H.E.: Marching Cubes a High Resolution 3D Surface Construction Algorithm, *Computer Graphics*, Vol.21, No.4, pp.163–169 (1987).
11) 小堀，石黒，久津輪：境界表現モデルからOctree表現への一変換手法，システム制御情報学会論文誌，Vol.8, No.3, pp.97–105 (1995).

(平成7年 2 月 22 日受付)
(平成7年 10 月 5 日採録)

米川 和利（正会員）

1971年3月15日生．1995年大阪工業大学大学院工学研究科修士課程修了．同年（株）日立製作所に入社し，現在に至る．大学院にて三次元CADシステムに関する研究に従事．現在はシステム，ソフトウェア開発に従事．工学修士．

小堀 研一（正会員）

1951年生．1975年山梨大学大学院修士課程修了．工学博士（大阪府立大学）．1975年シャープ（株）に入社．以後，一貫してCAD・CAM，CGに関する研究開発に従事．1991年大阪工業大学に奉職．現在同大学工学部電子工学科教授．CAD，CG，バーチャルリアリティの研究に従事．著書「CAD，CG基本用語集」（工業調査会），「三次元CG」（オーム社）など．システム制御情報学会，精密工学会，日本設計工学会，ACM各会員．

久津輪敏郎（正会員）

1941年生．1966年大阪工業大学電子工学科卒業，1973年大阪府立大学大学院電子工学専攻博士課程修了．工学博士．同年大阪工業大学電子工学科講師，助教授を経て，1984年教授．現在に至る．確率順序機械，論理回路の故障診断，高速演算方式，論理回路の自動合成，電気電子用CADシステムなどの研究に従事．「論理回路工学」（共著）など．電子情報通信学会，プリント回路学会，IEEE各会員．